Shell Services International

Data Integration Architectures

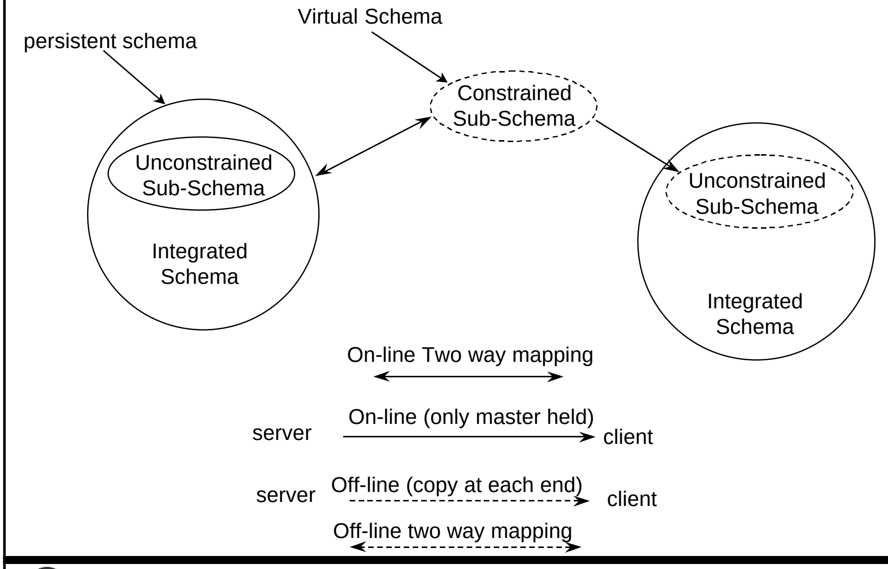
Some basic principles and possible configurations

Matthew West

***A**sset **I**nformation **M**anagement*

01/30/98

Basic Elements



persistent schema

Virtual Schema

Constrained Sub-Schema

Unconstrained Sub-Schema

Integrated Schema

Unconstrained Sub-Schema

Integrated Schema

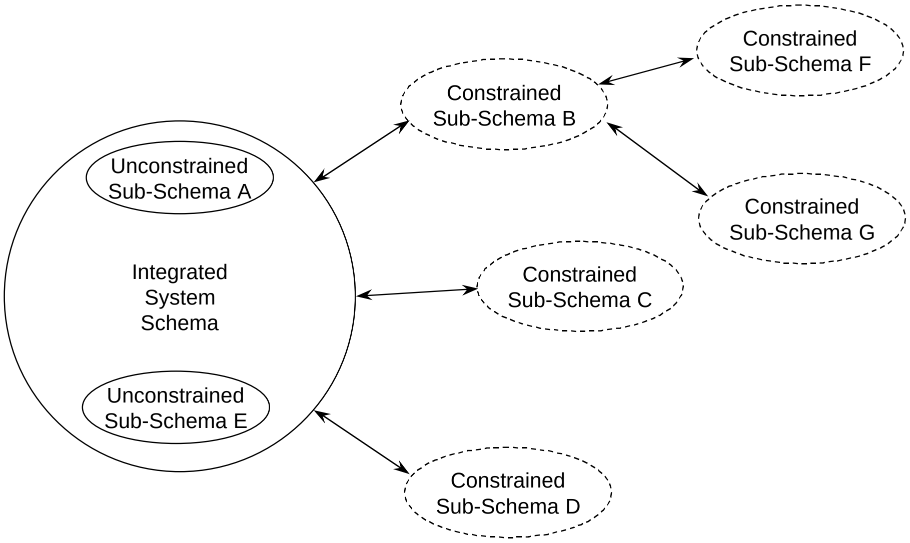
On-line Two way mapping

server On-line (only master held) client

server Off-line (copy at each end) client

Off-line two way mapping

Integrated System



Integrated System Schema

Unconstrained Sub-Schema A

Unconstrained Sub-Schema E

Constrained Sub-Schema B

Constrained Sub-Schema C

Constrained Sub-Schema D

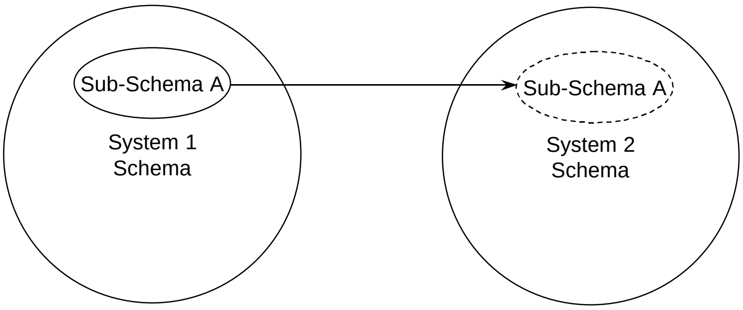
Constrained Sub-Schema F

Constrained Sub-Schema G

Notes

- The previous slide represents a traditional integrated system.
- Sub-schemas can be seen as like SQL views for particular functions.

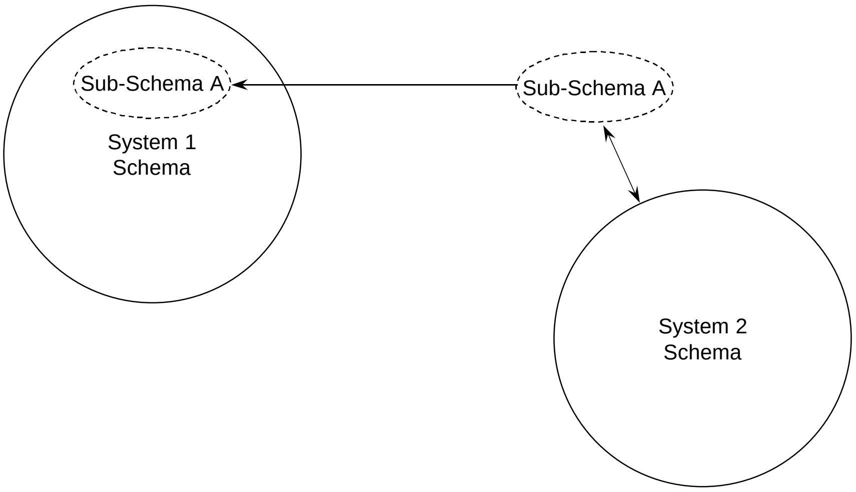
Tightly Coupled - Type 1



Notes

- This is essentially the “wrappering” of one application for some of its data.
- The access is dynamic and on-line

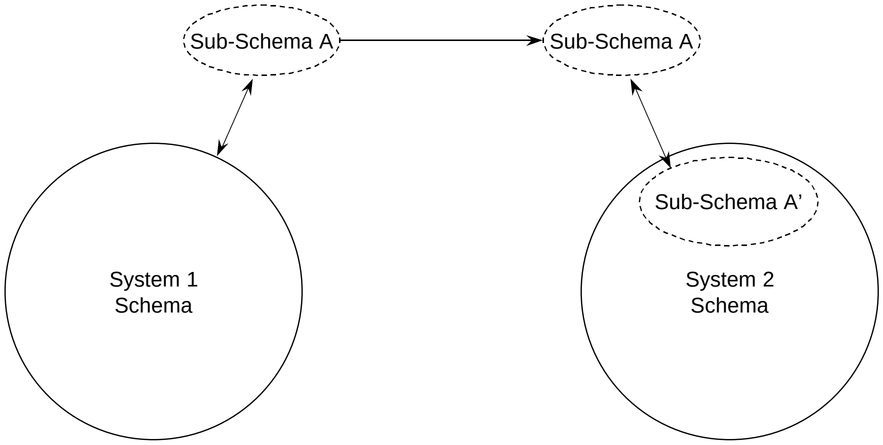
Tightly Coupled - Type 2



Notes

- A sub-schema of one application is defined as a view on another
- On-line access

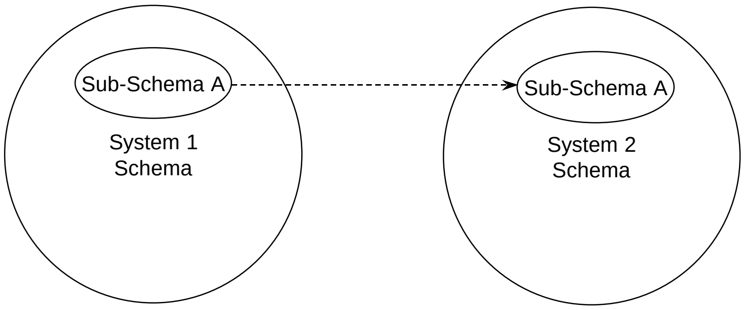
Tightly Coupled - Type 3



Notes

- Here it is a constrained view of one schema, which matches a constrained view of another schema.
- Hard to see why you would do this vs previous example unless you had to e.g. each dealt with an extended subset of the others data in the area of overlap.

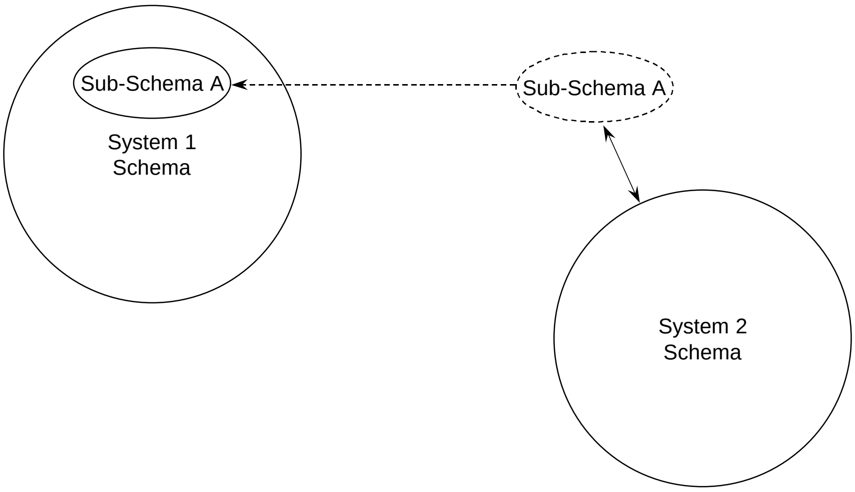
Loosely Coupled - Type 1



Notes

- Here data exchange is used with controlled data duplication, instead of on-line update.
- An unlikely case (except by design) when the two data models happen to match where they overlap.

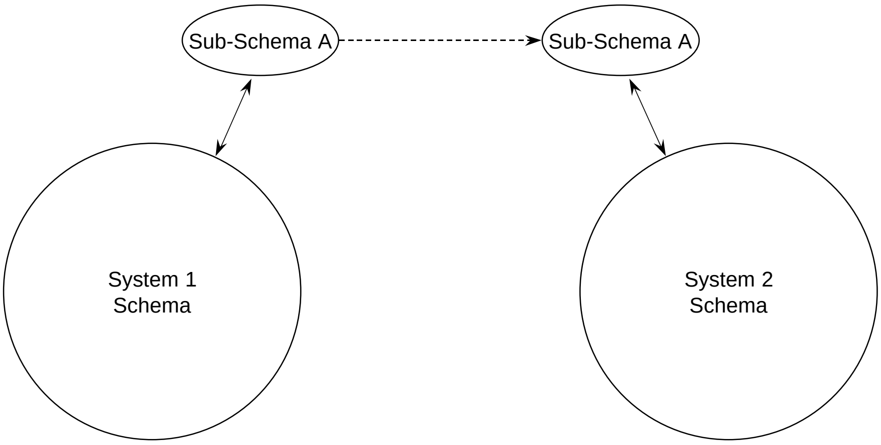
Loosely Coupled - Type 2



Notes

- A more general case where the server has a broader view than the client.

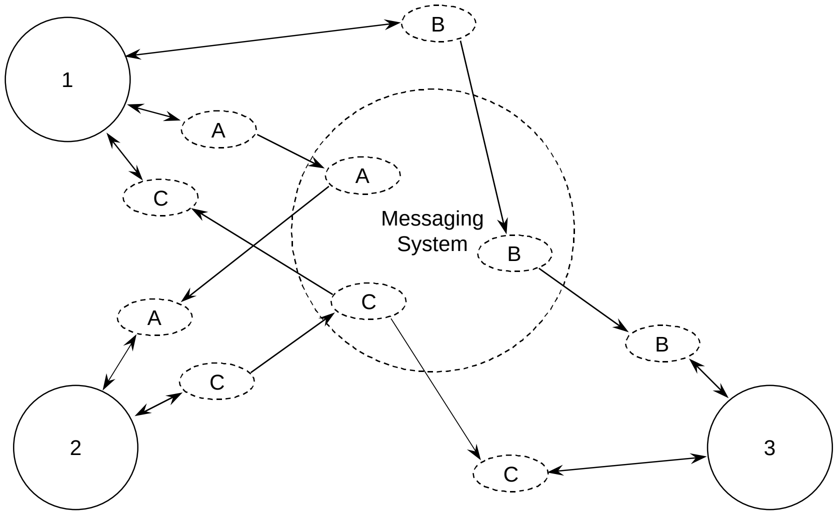
Loosely Coupled - Type 3



Notes

- Again the difficult case where each system deals with an extended subset of the information in the area of overlap
- Data Exchange

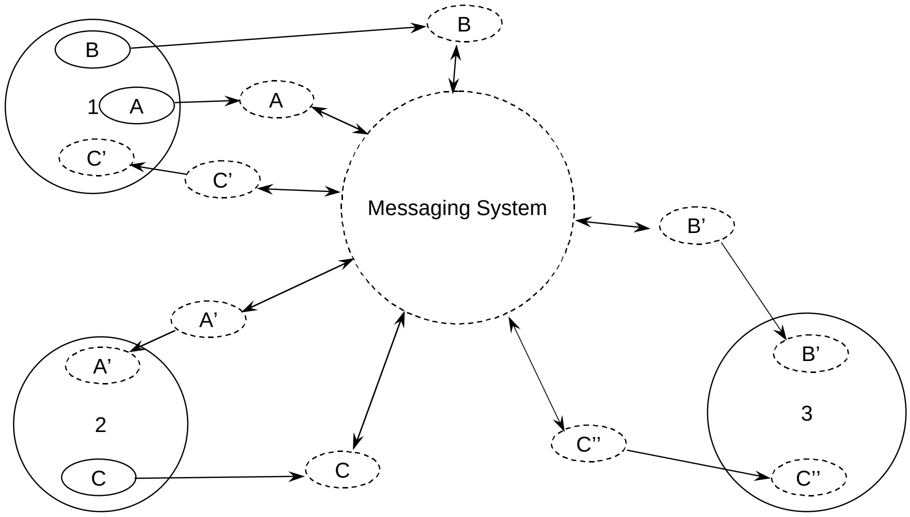
On-line Messaging (SDAI)



Notes

- This messaging system puts most of the intelligence with the applications.
- The result is the duplication of data model mapping expertise and functionality.
- None-the-less suitable when the applications are owned and operated say by different companies

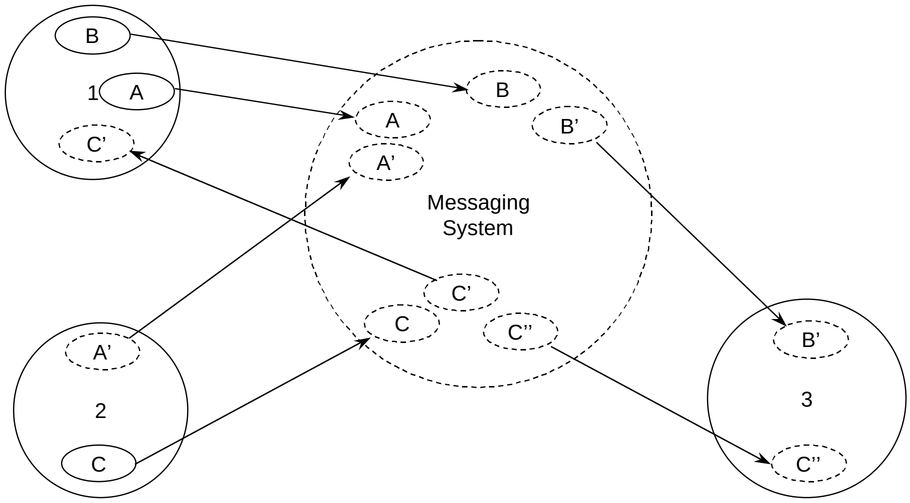
Federated Databases



Notes

- This messaging system has a central model and external mappings to each system
- The complexity is managed centrally, rather than being replicated for each application

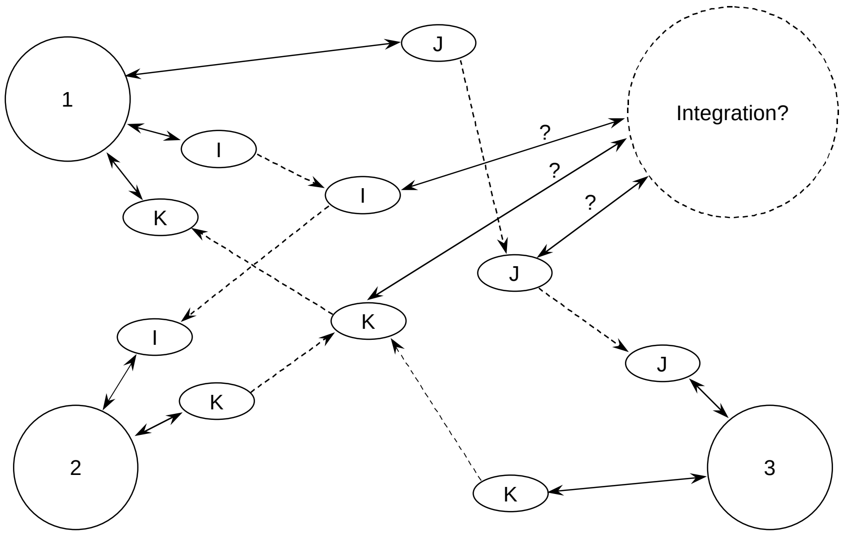
On-line Messaging System 3



Notes

- Here the mapping is integral to the messaging system model.
- External interfaces are simple.
- Complexity is managed through the data model itself.

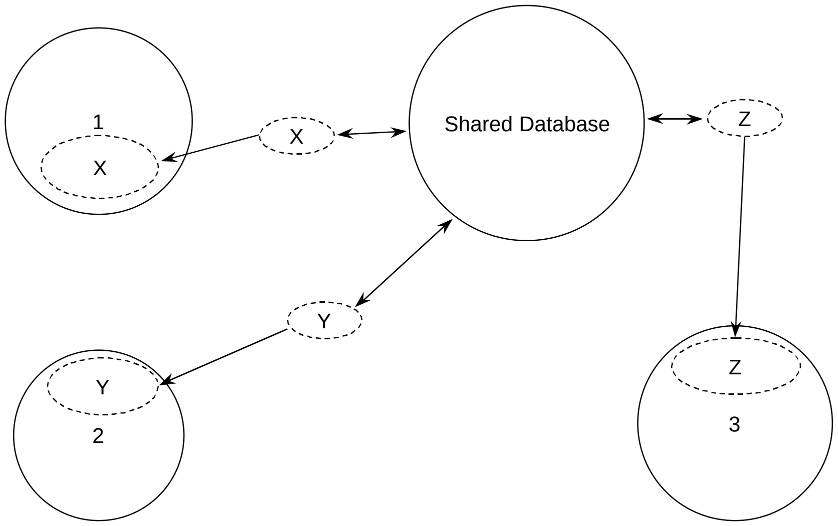
STEP Exchange Architecture



Notes

- This is an attempt to characterise the STEP data exchange architecture.
- ?? indicates where the integration doesn’t always work.

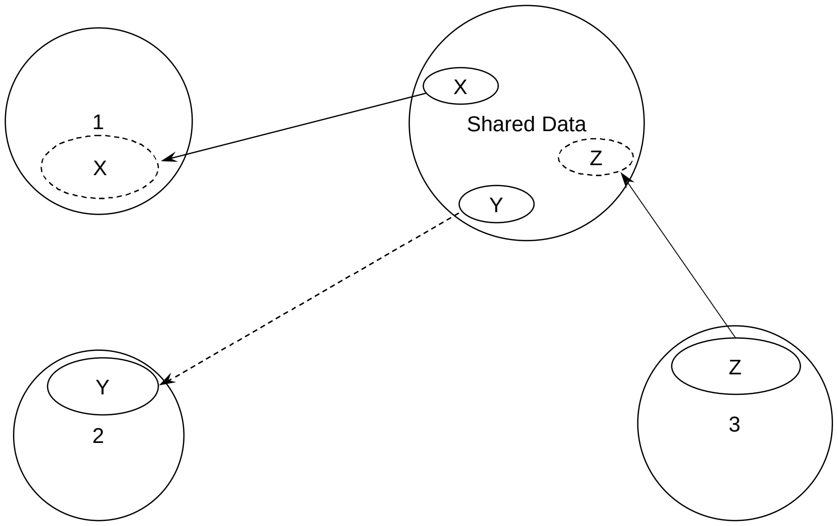
Shared Database



Notes

- This option shows a simple shared database as the master for shared data.
- Data is made available through application specific views

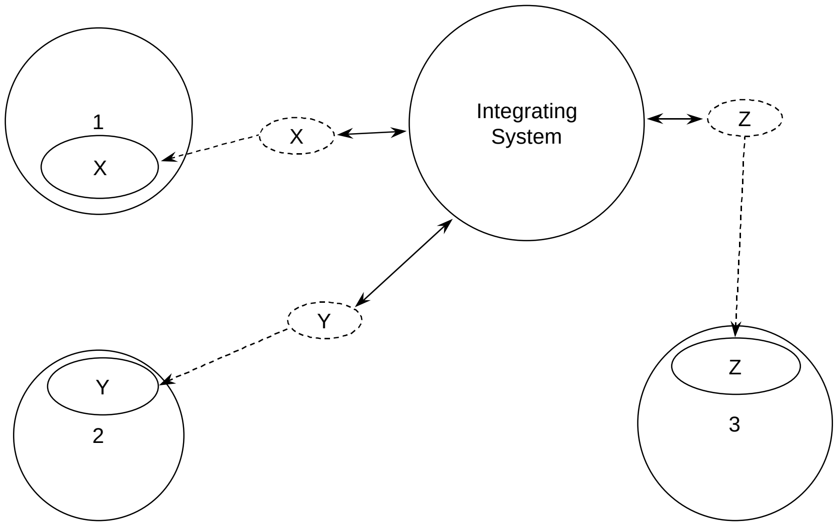
Proposed SC4 Data Architecture



Notes

- This is my current thinking for how the SC4 Data Architecture should handle integration.
- All mappings are handled by a single central data model.
- This incorporates views that match application models for shared data
- Persistence can be managed either in a central database for shared data, or by messaging from source applications.

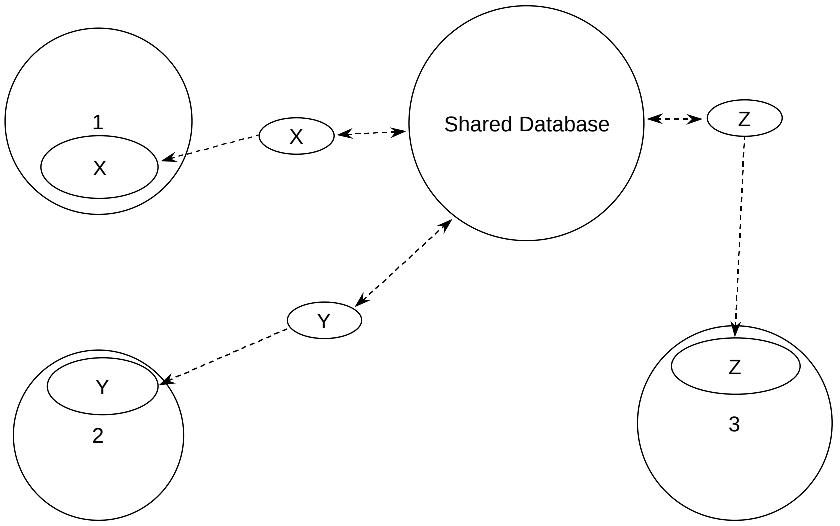
Loosely Integrating Database



Notes

- Integration through controlled data duplication
- Mapping is done dynamically on demand for a data set
- Mappings managed centrally

More Loosely Integrating Database



Notes

- Shared data is output to a mapping system that converts the data before exchange takes place.
- The mapping system could be common, and data model driven.
- The shared database could represent only temporary storage for integration purposes.